Lightweight Method for Detecting Fake Authentication Attack on Wi-Fi

Muhammad Yusuf Bambang Setiadji
Cyber Security Engineering
Sekolah Tinggi Sandi Negara
Bogor, Indonesia
yusuf.setiadji@stsn-nci.ac.id

Ramadhan Ibrahim
Badan Siber dan Sandi Negara
Jakarta, Indonesia
ramadhan.ibrahim@bssn.go.id

Amiruddin Amiruddin
Cyber Security Engineering
Sekolah Tinggi Sandi Negara
Bogor, Indonesia
amir@stsn-nci.ac.id

***Abstract*— Wireless networks, despite providing better access and flexibility, have vulnerabilities that are easier to realize compared to wired networks. Fake authentication attack can be taken by an attacker prior to carrying out a Man in the Middle attack to intercept the other party's communication. Such an attack is generally carried out in public places that provide free Wi-Fi access. Detection of fake authentication is necessary to maintain communication success. Several methods have been applied to detect fake authentication such as the use of Wireless Intrusion Detection System (WIDS) or certificates on Transport Layer Security (TLS). However, attackers can trick the use of WIDS or TLS. Moreover, the WIDS and TLS techniques require large costs and computations. In this study, a lightweight method based on the comparison of BSSID/MAC address for detecting fake authentication is proposed. The lightweight method is implemented by creating an application that runs on Android mobile phones, and Linux operating system. We compared the detection performance of the device with the proposed application and the one without the proposed application. It can be concluded that the proposed method using comparison of BSSID / MAC address is an effective way to detect fake authentication attacks on Wi-Fi networks.**

Keywords—fake authentication, fruitywifi, karma, Wireless Fidelity

I. INTRODUCTION

Wireless Fidelity, abbreviated as Wi-Fi, is one way to connect computing devices to the wireless network. With this wireless connection, Wi-Fi has advantages that cannot be provided by cable networks, such as convenience and flexibility of use. Not surprisingly, this network connection is very popular used in public places (hotels, coffee shops, restaurants), as well as private places (residences, offices) [9]. However, the convenience of using Wi-Fi also makes the emergence of fad or crime to abuse it, for example stealing user data or information. Attackers install fake Access Point to trap users and then steal user identity information. The information obtained can then be used to carry out serious attacks, for example carrying out a Man in the Middle (MITM) attack to intercept communications that occur on computer networks. However, to apply such an attack, the attacker must be part of the telecommunications system used by the victim [1] [11].

In Wi-Fi networks, attackers can become part of the telecommunications system by exploiting the weaknesses of probe request frames and probe response frames at the stage of the Wi-Fi association [2]. The attacker will monitor the probe request frame, which is sent automatically by the target, and extract the Service Set Identifier (SSID) value. The SSID value is then used as a pseudonym for the Access Point intended for the victims. To monitor probe requests, however, compatible hardware and software are needed. In this study, **karma** [3] which runs on the Debian operating system is used to conduct fake Access Point. Other than monitoring, karma also functions as a honeypot hotspot to attarct the victim's Wi-Fi devices to associate with a device that karma runs on. After the victim's devices is associated, the attacker has become part of the telecommunication system and can tap all communication from and to the victim's devices. To overcome this problem, as main contribution of our work, this research proposed a lightweight technique for detecting fake authentication based on Basic SSID (BSSID) / MAC address verification.

This paper is organized according to the general structure as follows. Section 1 describes the background that stirr the motivation for the research; Section 2 summarizes the related studies on Wi-Fi attacks; the proposed method is explained in Section 3; Implementation and discussion are given in Section 4; and Section 5 concludes the study.

II. RELATED WORKS

A fake authentication attack on Wi-Fi aims to intercept communications made by two entities. There are several previous studies with similar themes, i.e. detection of MITM attacks on Wi-Fi. In this section some studies related to MITM attacks and the weaknesses of association / authentication on Wi-Fi are explained.

Authors in [4] detected Rouge Access Point (RAP) which is a security threat for organizations / institutions if its use is not controlled. In addition, RAP can also be an entry point for MITM attacks. The detection of RAP is done at the edge of the network, which is on the side to the public internet point. Detection is done using the packet analyzer functionality, **tcpdump**. By carrying out that analysis, it can be distinguished between official WLAN networks and unofficial WLAN networks (RAP). Weakness of this technique is that the tool is made only for laptop devices, not for mobile devices.

Authors in [5] studied and detected Stealth MITM attacks by using the Wireless Intrusion Detection System (WIDS). Stealth MITM attacks can occur because the attacker can directly modify the reply frame of Address Resolution Protocol (ARP). The attacker can then exploit the weaknesses on key management of Wi-Fi Protected Access version 2 (WPA2). By providing / adding WIDS

along with the ARP, any Stealth MITM can be proven detected. However, the detection process requires additional device that has consequences for the need for additional cost.

Another variant of RAP, which exploits the Wi-Fi associaton and authentication is called Evil Twin attack. Research in [6] [12] [7] suggests various ways of detecting if an evil twin is present. While all three research claim to have succeeded in detecting evil twin, each have assumptions that limits its use. Author in [6] needs a server and assumes that a certain malware application be present in the victim before the attack. While the research in [12] uses Received Signal Strength Indicator (RSSI) specifically for detecting evil twin in a smart home environment. Author [7] argues that a fake, software-based access point will exhibit substantial difference in action or response if compared to a real, hardware based one. The solution offered is based on differentiating these exhibitions. Unfortunately, author [7] does not provide clear direction on implementing the solution.

The practice of Denial of Service (DoS) attacks are an auxillary to Man in The Middle (MiTM) attacks, because it can disconnect a subscriber and reconnects it to a RAP. The work in [2] proposed a technique to detect DoS attack in the wireless link layer. It is concluded that by implementing Wireless Link Layer Attacks Detection Algorithm (WLLADA), the DoS attack attempt can be detected earlier.

III. PROPOSED METHOD

The proposed method is to build a fake authentication detection application based on the comparison of BSSID / MAC addresses. The application was built following the standard methodology according to software development lifecycle which consists of needs analysis, design / modeling, development, and testing. Detailed explanation of these stages is given in the following section.

A. Need Analysis

The development of the application consists of functional and non-functional requirements. Needs analysis is done by analyzing documents/references related to user authentication [8]. From the document, identified needs are as follows:

1. *Functional Needs*
   - The application can get / capture SSID and BSSID value of the Access Point;
   - The application has the ability to generate hash values using certain hash function algorithm;
   - The application is able to store the computational results of the hash function;
   - The application can make a comparison between the SSID and BSSID values that have been stored and the SSID and BSSID values obtained / captured in real time.
2. *Non-Functional Needs*
   - The application can run on the Android and Linux operating systems;
   - The application uses the Python programming language desktop applications and Java-based Android for mobile-based applications;
   - The application runs when the device is turned on, and raises a warning if it detects an attack.

B. Threat Model

To further understand the threat that fake authentication has on the end users, a threat model as shown in Fig. 1 was used to alow the attakcer launch a fake authentication attack. According to [10], the attack can happen if a subscriber had associated with a Wi-Fi access point, shown as number "1" in Figure 1. If the subscriber saves the associated Wi-Fi access points to subscriber device and enables the "automatic connection" option, then it is vulnerable to fake authentication. The fake authentication happens if said subscriber goes out of reach of his/her genuine access point, and searches (broadcasts Probe Request frames) for available Wi-Fi options. The fake access point in noticing this Probe Request, will respond masquerading as the genuine access point, shown as number "2" in Figure 1. The subscriber (now victim) is associated to a fake access point, and connects to the internet through it.

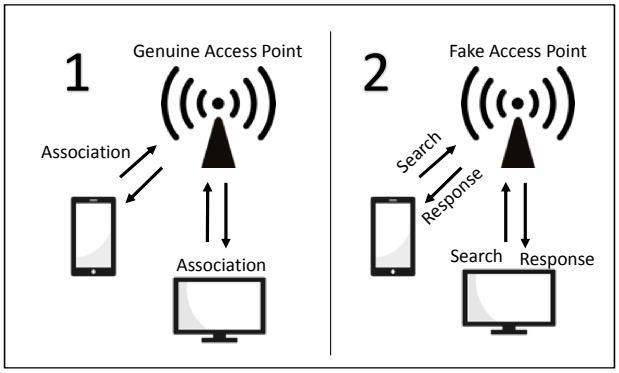

Fig. 1. Threat Model

C. Design/Model

The design/model shown in Fig. 2 shows the general work process of the proposed application. The detailed description of each step of Fig 2. is given as follows:

1. The device will associate with a Wi-Fi network Access Point;
2. The application acquires the SSID of the Access Point;

```
SSID = extract SSID (1)
```

3. The application extracts the BSSID of the Access Point;

```
BSSID = extract BSSID (2)
```

4. The application checks the connection type by looking for the SSID in the database.
   a. If the connection has been previously associated (the SSID is found in the database), go to step 7.

```
if checkSSID(SSID) == 1 (3)
```

   b. Otherwise, it is a new connection (SSID is not found in the database), go to step 5;

```
if checkSSID(SSID) == 0 (4)
```

5. The application hash the extracted-BSSID using a hash function;

```
HBSSID = hash(BSSID) (5)
```

6. The hashed-BSSID along with the SSID are stored in the database, then go to step 10;

```
store(SSID) (6)
store(HBSSID) (7)
```

7. The application hash the extracted BSSID and then compared it to the hash function value (from the same SSID) stored in the database.

```
H2BSSID = hash(BSSID) (8)
if H2BSSID =/= HBSSID then disconnect (9)
if H2BSSID == HBSSID then connect (10)
```

   a. If the result is not the same, then a fake authentication attack is assumed in progress, go to step 8.
   b. If the comparison result is the same, go to step 9;
8. The application will disconnect from the access point and give a warning to the user about a fake authentication attack;
9. The application responds that the connection is secure and does not interfere with the Wi-Fi connection;
10. Finish

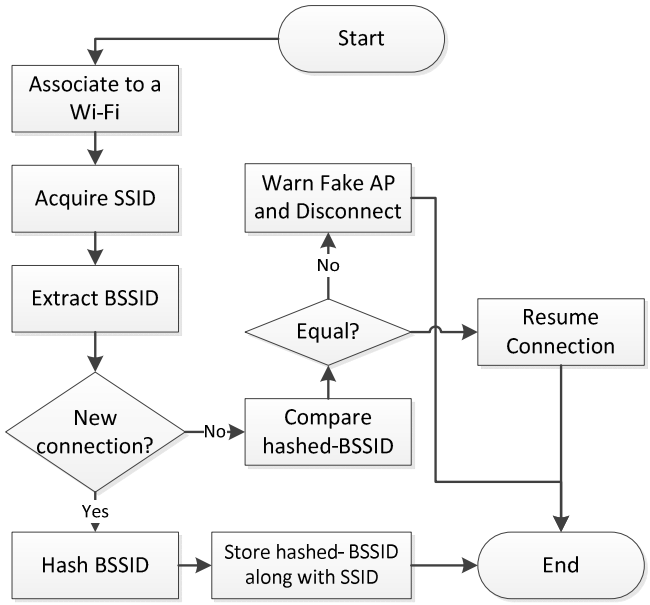

Fig. 2. Flowchart of the proposed method

IV. Implementation and Testing

A. Implementation

The proposed application was developed on two platforms, desktop and mobile. The implementation of desktop application uses the Python programming language whereas the mobile application using Java Android. The main functions of the application, **extract BSSID** and **Compare hashed-BSSID,** are implemented in both platforms and explained as the following:

a) Extract BSSID

For the desktop-based application, **extract BSSID** is implemented by the following source

```
def getBSSID():
      BSSID   =   subprocess.getoutput\
("/sbin/iwconfig         wlan0").split\
("\n")[1].split()[5][0:]
      return BSSID
```

whereas for mobile-based application, it is implemented through the following source code

```
wifiManager =
(WifiManager)getApplicationContext()/
.getSystemService(Context.WIFI_SERVICE)
;
  connection =
wifiManager.getConnectionInfo();
  tSSID = connection.getSSID();
  tBSSID = connection.getBSSID();
```

b) Compare hashed-BSSID

This function is implemented on desktop-based application through the following source code

```
def ceklistBSSID():
    hashing2 = hashBSSID()

    if            hashing2            in
open('/home/snake/Desktop/savemac.txt')
.read():
        print("BSSID is present")
        os.system("clear")
        print("disconnecting")
        os.system("echo password | sudo
-S ifdown wlan0")
        loading()
        gui.vp_start_gui()
        main()
    else:
        savessid()
```

whereas for mobile based application, it is implemented through the following source code

```
public void cekBSSID(String BSSID){
   Toast.makeText(this,"saat
ini:\n"+tBSSID,Toast.
   LENGTH_LONG)
     .show();
     if(BSSID.equals(tBSSID)){

Toast.makeText(this,"SAFE",Toast.LENGTH
_LONG).
       show();
     }else{

Toast.makeText(this,"MITM",Toast.LENGTH
_LONG)
       .show();
         }
     }
```

B. *Testing*

Application Testing is done using Unit Testing and System Testing. Unit testing is carried out on each function separately to ensure that the functionality of the function has been fulfilled and the test results are summarized in Table I. The system testing is done by testing the system as a whole and then comparing it with the functional requirements that have been previously set and summarized in Table II. The results of the two tests indicate that the functions stand alone, and the overall system can run as expected.

TABLE I. UNIT TESTING

No.	Function Name	Unit Testing Results
1.	get_net_info( )	Correct
2.	getssid( )	Correct
3.	getBSSID( )	Correct
4.	hashBSSID( )	Correct
5.	savessid( )	Correct
6.	saveBSSID( )	Correct
7.	ceklistBSSID( )	Correct
8.	loading( )	Correct
9.	main( )	Correct
10.	p3keccak.Sha3_512()	Correct

TABLE II. SYSTEM TESTING

No.	Functional Requirements	Results
1.	Application can retrieve SSID and BSSID/MAC Address	Fulfilled
2.	Skema dapat melakukan komputasi fungsi *Hash* terhadap nilai BSSID/ MAC *Address* Application can process BSSID/ MAC Address with a hash function	Fulfilled
3.	Application can store the results of the hashed BSSID/ MAC Address	Fulfilled
4.	Application can compare the stored hashed BSSID/ MAC Address to a real time acquired hashed BSSID/ MAC Address	Fulfilled

V. FAKE AUTHENTICATION DETECTION TESTING

A. *Testing parameters*

TABLE III. EXPERIMENT ENVIRONMENT PARAMETERS

Parameters	**Values**
Network	WLAN
Victims	2 victims, V_1 and V_2 V_1 using laptop with Linux OS V_2 using hand-phone with Android OS
Attacker	Conduct fake authentication
MAC Addr of V_1	5C:93:A2:FE:CF:9B
MAC Addr of V_2	54:DC:1D:60:5A:E2
MAC addr of Attacker	00:08:CA:58:64:97
Assumption	All two victims have previously associated with other Access Point before the test

Application testing is carried out with experimental environment parameters as summarized in Table III. The network used is WLAN. There are two victims, V_1 and V_2. V1 using a laptop with OS Linux with the device's MAC Address is 5C: 93: A2: FE: CF: 9B while V2 using a mobile with an Android OS with the device's MAC Address is 54: DC: 1D: 60: 5A: E2. A fake authentication attack is carried out by an attacker. With the MAC Address of the device 00: 08: CA: 58: 64: 97. At the time of testing, it was assumed that the two victims V_1 and V_2 used a device that had been associated with another Access Point

B. *Testing implementation*

Fake authentication attack was conducted by attacker through the following steps

1. Turn on the **fruitywifi** application;
2. Activate module [AP], karma, and nginx as shown on Figure 3.

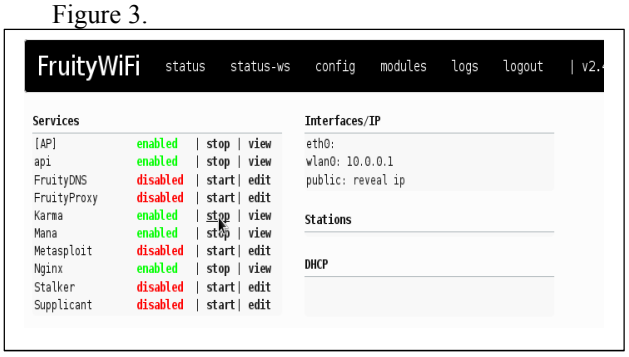

Fig. 3. Fruitywifi Setup

After the configuration setup as on Fig. 3, the fake authentication attack ran. For making sure, a list of victims associated to the **furitywifi** appeared on the display as shown on Figure 4.

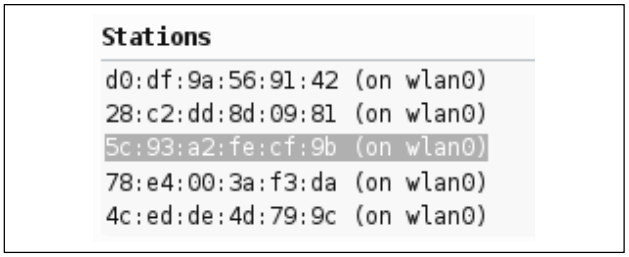

Fig. 4. Fake Authentication Attack

In the experiment, the victim's laptop and cellphone that does not have a BSSID detection application will immediately associate with a fake Access Point run by fruitywifi. The association can be proven in Fig. 4 and Fig. 5 for laptops, and Fig. 6 and Fig. 7 for cellphones.

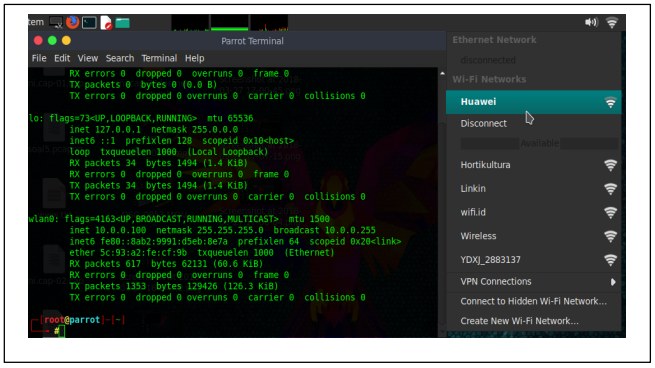

Fig. 5. Victim's laptop

In Fig. 4 the MAC address of the victim laptop, 5C: 93: A2: FE: CF: 9B, is detected in the fruitywifi application as one of the associated devices. In Fig. 5 the victim laptop does not display anything out of the ordinary, even the name of the access point remains the same as the one previously associated, namely "Huawei".

For attacks using mobile phones, the conditions are not much different from attacks on laptops. As seen in Fig. 6 the victim's cellphone is directly associated and doesn't detect anything suspicious when associated with the intended fake Access Point. In the **fruitywifi** application shown in Fig. 7, the MAC address of the victim's cellphone is detected as one of the associated devices.

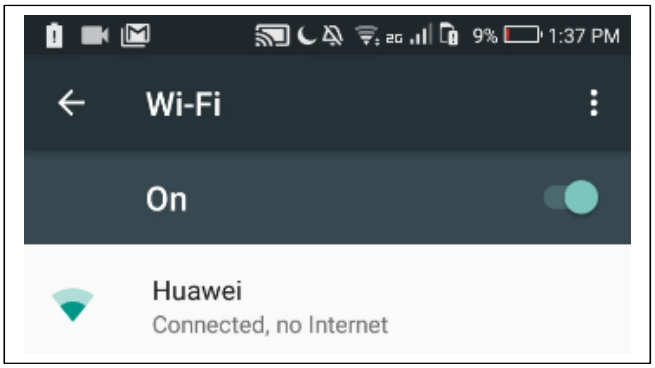

Fig. 6. Victim's cellphone

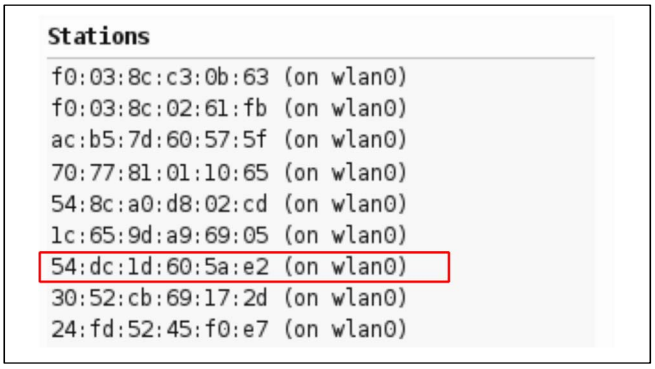

Fig. 7. Victim's cellphone detected

From the proof of the fake authentication attack, it can be concluded that the attack was successfully carried out by the attacker when the BSSID / MAC address detection application was not executed on that device. The next step that can be done by the attacker is tapping the communication (passive attack), or changing the communication that is currently running (active attack). Both of these attacks are dangerous and are a serious threat to victims.

Fake authentication attacks will not occur if the victim device is able to detect changes in the MAC address of the SSID that has been associated. Fig. 8 and Fig. 9 each illustrate the results of the implementation of a fake authentication application if it is run on the victim's laptop and cellphone during a fake authentication attack. Looking at the details in Fig. 9 it can be seen that the MAC address detected is the attacker's MAC address, but the SSID displayed is "STSN-AP42".

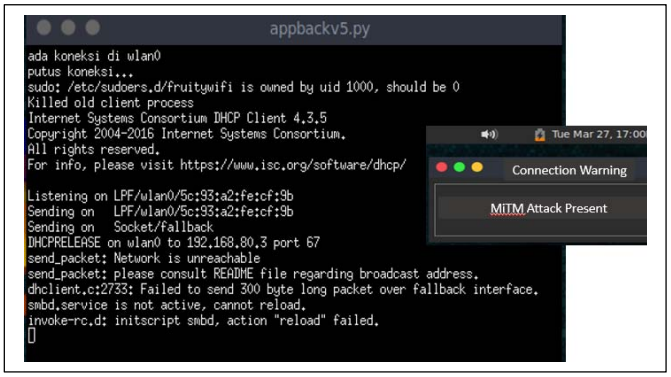

Fig. 8. Disconnection and warning on victim's laptop

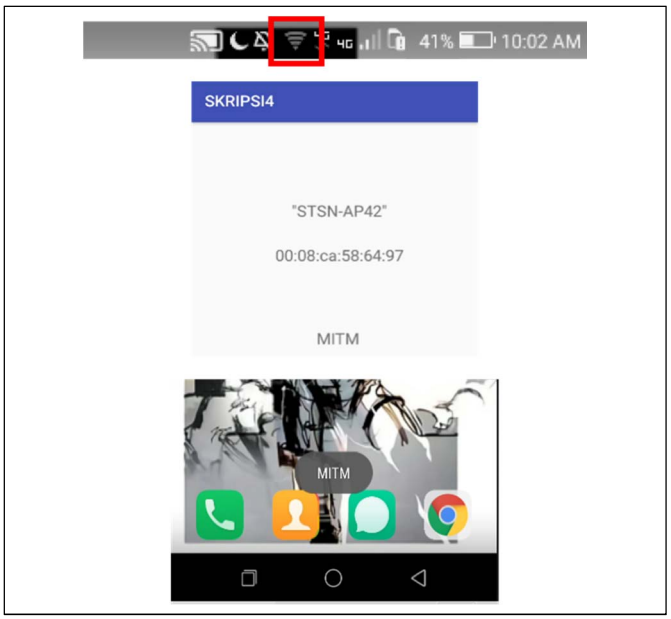

Fig. 9. Disconnection and warning on victim's cellphone

TABLE IV. EXPERIMENT SUMMARY

No.	Condition	With Application	Without Application
1	Laptop Wi-Fi associated with new access point	Connection established	Connection established
2	Laptop Wi-Fi associated with known access point	Connection established	Connection established
3	Laptop Wi-Fi associated with known SSID in a fake authentication attack	**Connection Denied**	Connection established
4	Cellphone Wi-Fi associated with new access point	Connection established	Connection established
5	Cellphone Wi-Fi associated with known access point	Connection established	Connection established
6	Cellphone Wi-Fi associated with known SSID in a fake authentication attack	**Connection denied**	Connection established

The experiment results are summarized in Table IV. It can be seen that the user's device, either laptop or cellphone that tried to connect to an SSID in a fake Access Point, will immediately disconnect the connection. This happened

because the application of the MAC address comparison installed on the devices detected a change in the SSID that means that the SSID was incompatible with the data stored in the database.

VI. CONCLUSION

In this research, fake authentication attack has been implemented by installing fake Access Point to trap victims' devices in the form of laptops and cellphones. Then, to detect and prevent these fake authentication attack, a lightweight detection method based on comparison of BSSID / MAC addresses is proposed and implemented. The proposed method was implemented in applications running on the Linux and Android operating systems.

Compared to research done by [5] and [6], our detection mechanism does not need the presence of additional apparatus to detect a rouge access point. Furthermore, our solution can detect MITM attacks in public network, while research done in [4] and [12] only applies to certain confined networks, that are under the control of the user.

Based on tests carried out using karma application, the fake authentication attacks were successfully carried out and the victims' devices successfully associated to the intended fake Access Point. Using the MAC address comparison application, whenever the application detects an inequality on the MAC address comparison, it will automatically disconnect the Wi-Fi connection since it assumes it as a fake authentication attack. In addition, the application does not interfere with other legitimate connections that occur while detecting and disconnecting the fake connection. Thus, it can be concluded that the application meets the purpose of detecting fake authentication attacks on Wi-Fi.

REFERENCE

[1] B. Aziz and G. Hamilton, "Detecting Man-in-the-Middle Attacks by Precise Timing," in Third International Conference on Emerging Security Information, Athena, 2009.

[2] M. A. C. Aung and K. P. Thant, "Detection and Mitigation of Wireless Link Layer Attacks," in IEEE SERA, London, 2017.

[3] R. Wood, "Projects: Karma," [Online]. Available: https://digi.ninja/karma/. [Accessed 6 May 2019].

[4] S. Shetty and M. Song, "Rogue Access Point Detection by Analyzing Network Traffic Characteristics," in IEEE Military Communications Conference, Orlando, 2007.

[5] V. Kumar, S. Chakraborty, F. A. Barbhuiya and N. Sukumar, "Detection of Stealth Man-In-The-Middle Attack in Wireless LAN," in IEEE International Conference on Parallel, Distributed and Grid Computing, Solan, 2012.

[6] Vineeta Jain, Vijay Laxmi, Manoj Singh Gaur, Mohamed Mosbah, "ETGuard: Detecting D2D Attacks using Wireless Evil Twins," Computers & Security, 2019.

[7] Fabian Lanze, Andriy Panchenko, Ignacio Ponce-Alcaidey, Thomas Engel, "Hacker's Toolbox: Detecting Software-Based 802.11 Evil Twin Access Points," 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, 2015.

[8] A. J. Evans, W. Kantrowitz and E. Weiss, "A user authentication scheme not requiring secrecy in the computer," Communications of the ACM, vol. 17, no. 8, pp. 437-442, 1974.

[9] V. Roth, W. Polak and E. Rieffel, "Simple and Effective Defense Against Evil Twin Access Points," in WiSec'08, Alexandria, 2008.

[10] D.A. Dai Zovi and S.A. Macaulay, "Attacking Automatic Wireless Network Selection," in Sixth Annual IEEE SMC Information Assurance Workshop, West Point, 2005.

[11] Shao-Long WANG, Jian WANG, Chao FENG and Zhi-Peng PAN, "Wireless Network Penetration Testing and Security Auditing," in 3rd Annual International Conference on Information Technology and Applications, Hangzhou, 2016.

[12] Zhanyong Tang, Yujie Zhao, Lei Yang, Shengde Qi, Dingyi Fang, Xiaojiang Chen, Xiaoqing Gong, and Zheng Wang, "Exploiting Wireless Received Signal Strength Indicators to Detect Evil-Twin Attacks in Smart Homes," Mobile Information Systems, 2017.